Efficient Attribute-based Data Access in Astronomy Analysis

B. MA[1], A. Shoshani[1], A. Sim[1], K. Wu[1], Y. Byun[2], J. Hahm[3], M.-S. Shin[4]

[1]Lawrence Berkeley National Laboratory, USA
[2]Yonsei University, Korea
[3]Korea Institute of Science and Technology Information, Korea
[4]University of Michigan, USA

***Abstract* - Large experiments and high-performance computer models generate many petabytes of data. While Cloud Computing systems may meet the needs for analyzing these petabytes by harnessing the computing power of many distributed computers, the key challenge in effectively utilizing such a distributed system is the data management process, including storage, indexing, searching, accessing, and transferring data. Most analysis tasks perform computations on a subset of a large data records satisfying some user specified constraints on attribute (variable) values. This subsetting procedure is extremely important in that it reduces the network traffic to and from the cloud facilities. However, selected data records often span many different data files, and extracting the values out these files can be time-consuming especially if the number of files is large. This work addresses this challenge of working with a large number of files. We use a set of astronomical data set as an example and use an efficient database indexing technique, called FastBit, to significantly speed up the subsetting and thus optimize network usage. Overall, we aim to provide transparent and highly efficient attribute-based data access to scientists through a web-based Astronomy Data Analysis Portal. We will discuss the system design, and options for managing an extremely large number of files while minimizing network usage and latency.**

***Index Terms* - Astronomy analysis, Cloud Computing, fastbit, data search, astronomy data.**

I. INTRODUCTION

Large experiments and high-precision simulations are providing invaluable insight into scientific phenomena. A well-known challenge in these scientific endeavors is the mountain of data they produce; many experiments and simulations are already producing multi-petabyte of data and data volumes are growing exponentially. This data explosion exist in nearly all fields of science, including astrophysics and cosmology, high energy physics, material science, climate modeling, fusion, and biology, to name a few. Many of these analysis tasks require large computing capacity, and in some cases, the computing power required for analysis is as much as that needed for the original simulation that produced the data.

Cloud Computing may satisfy the analysis requirements by distributing the computational tasks. In many cases, Cloud Computing systems are also the more cost-effective solutions in comparison with traditional centralized approaches. Sharing data across the Cloud Computing facilities can be achieved through data transfers over high-speed networks. However, the key bottleneck in effectively utilizing Cloud Computing is usually the data management process, including storage, indexing, searching, accessing, and transferring the data. Scientific data records may have an extremely large size, an extremely large file count, or both. While there are various techniques for managing data movement, a major issue is the amount of the data being moved that is not necessary for the analysis. In distributed data-intensive analysis environments, this problem can significantly degrade the performance of analysis. In particular, network transport latency, along with data access latency, usually accounts for a substantial portion of time required for cloud-based scientific analysis. Thus, it is important to have methods that extract the necessary data records at the source, before transferring the desired subsets to the analysis facility.

Many analysis tasks perform computations on a subset of data records selected based on the attributes data values. For example, when analyzing an astronomy dataset, one might want to plot a set of light curve for objects in a particular patch of the sky. The data needed for this operation could be retrieved from data records satisfying certain range conditions on Right Ascension and Declination. This subsetting procedure reduces the amount of data to be transported to the Cloud Computing facilities and is therefore critical to the overall effectiveness of the distributed analysis system.

Selected data records often span many different data files. The analysis programs must know which data files include the selected records. Extracting the values out these files can be time-consuming especially if the number of files is large. In some cases, the selected data records have to be reorganized such as clustering them based on the time stamps. Even though such subsetting and reorganization functions are frequently required, they are not well supported by current scientific data management systems.

We describe the design and development of a generalized attribute-based unified data access service that provides transparent and highly efficient data-access mechanisms and optimizes network usage by reducing the data transport load

at the source. This requires a high-level coordination of data discovery, data selection, index generation, data access, and data delivery. We use the system for the Astronomical data analysis based on the cloud environment. Our approach is to provide a general-purpose service framework that will enable scientists to manage the data flows.

II. Background

A. Astronomy Data Analysis

Astronomy analyses face the difficult challenge of managing the distributed access to massive data sets. For instance, the Sloan Digital Sky Survey (SDSS) Data Release 7 (DR7) [1] consists of 64TB of imaging and spectroscopic data. The imaging data covers about 8423 square degrees of "legacy" sky, with information on roughly 230 million distinct photometric objects, and about 3240 square degrees of SEGUE sky, with about 127 million distinct objects. The spectroscopic data includes data from 1802 main survey plates of 640 spectra each, and cover 8200 square degrees. Another example of a large database is SuperWASP Data Release 1 [2], which consists of about 120 billion data points in 18 million unique light curves derived from 3.6 million images. This takes up 5.9TB of disk space for the light curves, and over 20TB for the raw images.

These are only two examples of many large databases already available in the astronomy community. The size of databases is certain to increase quite dramatically in the coming years, in all areas of survey astronomy. In order to maximize the science products from such huge databases, often located in remote centralized data centers, it is of utmost importance to provide users with a general tool, which efficiently searches for subsets of desired data entries.

In addition to the increasing size of the data, there is also a significant increase in the distance between data and application scientists, due to the large international collaboration where the massive amount of data is generated or collected in a few centralize data repositories. The concept of Cloud Computing has been adopted in astronomy research for tackling the problem of the data analysis over massive datasets. Although there are few examples of Cloud Computing for this purpose, the effective usage of the Cloud Computing has been proven in astronomy only if the relevant data can be easily and quickly accessed by the computer resources.

For this study, we chose the Northern Sky Variability Survey (NSVS) [3,4] dataset, which contains light curves for 16 million objects with typically 60-400 measurements per object. The NSVS is used by astronomers for analysis of patterns in the night sky in search of rare and transient astronomical events.

While this dataset takes about 100 GB of disk space, it is divided into over 16 million files. The large number of files has proven to be challenging for most file systems and scientific data management systems. To reduce data management complexity, these files are grouped by a common attribute into 638 tar balls (each combining multiple files with the TAR function), and seven key attributes of all files are collected into a plain text catalog file. However, searches were still far from optimized, particularly in the handling of timestamps. Each NSVS observation contains a timestamp and analysis tasks typically require data records from specific time ranges, but the timestamps cannot easily fit in the catalog file. Any query involving timestamps needs to read and process all 16 million files at least once, making such analysis tasks extremely time-consuming and even more so if data post-processing and over-the-network data transfer is involved. Our approach to resolve this issue is to extend the NSVS catalog to include the time range of observations in each file, thus enabling more fine-grained search and selective data extraction/read from the dataset. This approach not only filters the data and reduces the amount of bytes delivered to end-users for the analysis, but should significantly reduce the time needed for data access, I/O operations, and network delivery. The Attribute-based unified data access service provides such transparent and highly efficient data-access.

B. Astronomy Data Cloud Portal

The StarCloud portal is a web-based service which provides user interfaces for search and extraction of astronomy data, as well as customized computing resources for the data analysis through Cloud Computing. The portal consists mainly of the data service and the analysis service. In the data service, a user can have an access to various astronomy datasets such as NSVS. It has an easy-to-use interface, where a user can input search parameter values and check the search results, then download the selected subset of the data. In addition, a user can download the data subset from the analysis computing resource as a tar ball using a URL provided by the data service system. The StarCloud data service became significantly more efficient owing to the backend Attribute-based Unified Data Access Service.

The astronomical scientists need a variety of analysis tools, including open source community-delivered software and in-house programs for the analysis of astronomical data. However, there is a large cost in money and manpower for managing private computing clusters. It is still hard to use a public cluster with such various needs of analysis software. Cloud Computing can alleviate these issues by providing private computing environment for analysis of large scientific datasets in a cost-efficient manner.

The StarCloud analysis service provides various astronomical tools with the Cloud Computing service. Users can have a virtualized private computing cluster on demand, which is customized with a specific operating system, libraries and scientific computing software in order to run the analysis on the data subset selected by the data service. They can select an analysis tool from listed astronomical data analysis tools on the portal, which is stored in virtual machine images, then request a data analysis resource with specific parameters, for example, the number of computing nodes, memory size, storage size, and so on. The request is delivered to the Virtual Cluster Service, which creates a computing cluster configured by virtual machines. The Virtual Cluster Service is a high-level tool in front of OpenNebula, a widely-used cloud management software, to create and manage virtual cluster instances, not just virtual machine instances.

C. Challenges

The main goal of this case study is not just to efficiently select the subset of light curve data for a given sky location and for a given time range. Based on the timestamp attribute of individual measurements, we further select data entries sharing the same set of timestamps, i.e., observed at the exact same time by the survey instrument. This sub-grouping is very useful in identifying and estimating the significance of systematic biases, which might have prevailed at the time of the survey observations and have not been properly removed by the original data analysis process. If this is the case, then the apparent variability we see in the light curve data of a particular astronomical source may not reflect the intrinsic nature of the source. Such biases may also hide whatever temporal variation the sky objects really have and could have been detected by the survey.

The identification and removal of systematic bias is called "de-trending" and recent successful examples are given in references [8] and [9]. In order to run the de-trending process, the light curve data has to be searched for a subset of measurement entries that share the exact same series of timestamps because only these subsets share the same systematic and environmental biases that occurred at the specific time of observations. De-trending is difficult for a very large database because of the need to select the data entries that can be correctly compared with each other. The selection process can be extremely time consuming with conventional approach as one needs to read through entire light curve entries, comparing the timestamp of a measurement entry with other entries for all other stars in nearby sky locations.

Depending on the nature of the astronomical survey and the intended science analysis with the database, there can be several different sets of database challenges. We believe that the present case study is an important representative challenge for all astronomical time-series survey databases.

III. Attribute-based Unified Data Access Service

Searching and refining dataset efficiently for analysis and moving the selected data reliably to the analysis computing facilities are difficult tasks. The Attribute-based Unified Data Access Service (AUDAS) will accommodate and support such complex tasks. Clients will only need to define a dataset of interest using range conditions on variables that are familiar to them and specify the data attributes for the analysis.

In this NSVS case study, we 1) indexed the light curve observation timestamps so that data subsetting does not require reading the entire NSVS dataset, 2) reduced the searching time from $O(n)$ time to $O(\log N)$ time using an indexing tool, called FastBit, and 3) optimized the data access process through a series of benchmarks that simulate a production-environment.

A. Service Framework and API

The Attributed-based Unified Data Access Service includes four components: Data Selection, Data Index, Data Access, and Data Delivery. When an analysis component contacts the Astronomical Data Analysis Portal, the data discovery mechanism first searches for relevant datasets through the metadata catalog provided by the user communities. The qualified datasets are further refined based on the selection criteria and the data index in Data Selection and Data Index components. These refined subsets are extracted and delivered to the client analysis component through the Data Access and Data Delivery components. Each component may be accessed separately based on the client needs. Figure 1 shows the overall design of the system and the interface between the components. The Data Access service API is the means of communication between clients and the service components as well as between the individual components of the service framework. To allow maximum flexibility, a RESTful web service is implemented for the APIs.

B. Data Index

For attribute-based data selection, the Data Index component generates indexes for the dataset. The goals of the Data Index component are to provide fast search across the dataset, and to allow for fine-grained data selection process in order to eliminate the data not needed for an analysis, thus significantly reducing the time needed for network and I/O operations. The Data Index component has a flexible design, which can select index generation either dynamically or statically. In most cases, the index is created for the entire dataset once, and is used for data selection as long as the dataset does not change. The dynamic indexing technique reduces the amount of index generation time, especially when datasets are very large and are dynamically growing. In this paper, the original CSV-based dataset is used. The search performance is described in the next section.

FastBit is used in the Data Index component in this case study. FastBit [5] is an efficient database indexing technique to speed up data search with conditions on attribute variables. It implements compressed bitmap index techniques for attribute-based searching operations using an SQL-like query system, and was designed to take advantage of the read-only nature of scientific data to provide more efficient search operations. FastBit has been demonstrated in numerous US DoE scientific applications as able to significantly speed up attribute-based searching operations. The storage cost of the index is reasonable. In this case study, the original plain-text catalog file takes about 1.3GB, and the FastBit indexes take up 2.2GB.

Work has been done to also use another tool, called FastQuery as the Data Index component. The recently developed FastQuery [6] search system, based on FastBit, allows users to quickly search and access HDF5 and NetCDF files without the need to convert the user data into the native FastBit format.

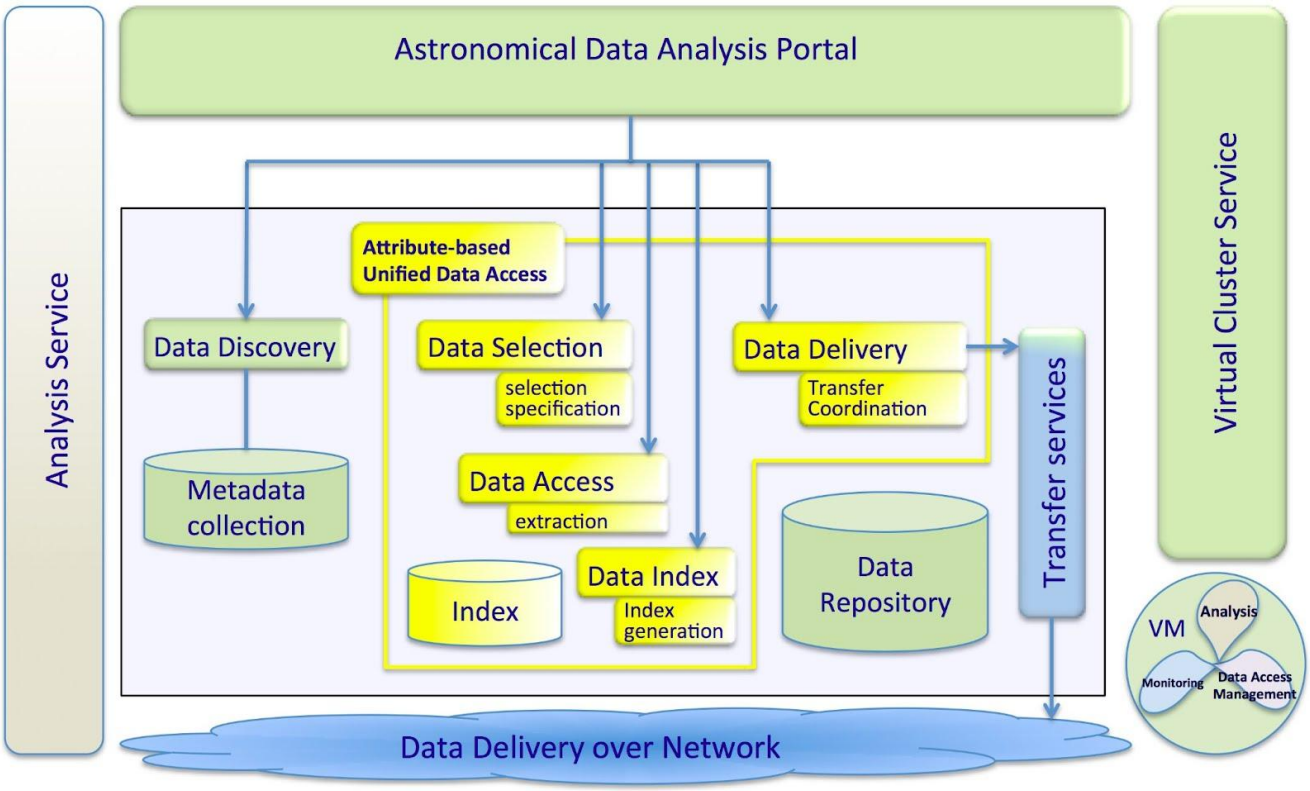


Figure 1: Overall architecture of the Attribute-based Unified Data Access Service

C. Data Selection

As the recipient of data search results, the Data Selection component is the entry point to the Attribute-based Unified Data Access Service. The role of this component is to collect attribute information based on the selection specification from the clients, validate the attributes, and select the data and filtering information from the metadata search results. The selection information is then returned to the client for the next steps in the data access. The selection specification can be generalized so that different datasets and analysis tools can be supported in the future. In our case study, SQL-like queries are provided to FastBit in order to search the NSVS according to specified attribute values.

D. Data Access

The Data Access component extracts files from the dataset based on the selection produced from the Data Selection component. Data filtering and post-processing can be customized for the dataset, and the final results are packaged into a designated archive to be delivered to the clients for the next steps in data analysis. Extensibility is built into the data filtering mechanism so that different data formats and analysis can be supported. In this case study, the appropriate files are extracted from the NSVS dataset based on the Data Selection results, and repackaged into a tar ball. Because this process is I/O intensive, a file system directory was mounted in memory (tmpfs) as the staging area for extraction, post-processing, and repackaging in order to reduce the processing time.

E. Data Delivery

The Data Delivery component provides a transparent way to access data based on supported transfer protocols. The Data Delivery component will coordinate the movement of the selected dataset to the destination, using the desired transfer protocol. As the initial scope of the project, HTTP-based data download is supported for the selected subsets to be transferred to the local destination. The extensibility is built into the component so that it would evolve with future networking infrastructures, supporting anonymous FTP, GridFTP [7] or SCP/SFTP.

F. AUDAS and Cloud-based Astronomy Data Analysis

Usage of the Attribute-based Unified Data Access Service (AUDAS) consists of two steps: general search based on the query string, and further filtering within the first search based on the time stamps. When a user submits an SQL-like query string to view and retrieve a subset of the NSVS data, the web service performs FastBit queries on the NSVS index in the backend, and retrieves, post-processes, and re-packages the data for users to download. When a user further specifies a time range to filter their initial search, AUDAS reads through every file of the pre-specified data subset, filters in light curve observations that fit in the time range, formats the output, and writes to a new set of files. Resulting data files are downloaded as a tar package file when requested.

A RESTful API has also been provided to the Data Access component of AUDAS to enable scriptable and

customizable data search and acquisition. Users enter the entire SQL-like query string and time ranges if desired as the fields to the URL query, and automatically download a tar ball as a REST response. The RESTful API ensures a clean and flexible separation of components of AUDAS.

IV. Results

A. Data Indexing

To enable more efficient use of the catalog from NSVS, we expand it to include the time range and then build a set of FastBit indexes. To build the FastBit indexes, we convert the text version of the catalog file into the binary format used by FastBit first and then use a FastBit command line tool to generate the appropriate bitmap indexes based on the characteristics of the data. On the set of NSVS data, converting the catalog file from text format to FastBit binary format took about 166 seconds, and generating the bitmap indexes took about 48 seconds. Since these are one-time operations, they do not factor at all into search time.

B. Data Searching

FastBit search was benchmarked against a line-by-line search of the NSVS plain-text catalog. This full catalog search required an average of 138s, while the same search using FastBit indexes took a mere average of 0.121s, a speedup of over 1100. On a cold start of the FastBit program, in which the large bitmaps have not been cached to memory yet, startup is only around 4x slower than average, about 0.513s, but still is hundreds of times faster than the linear string search.

C. Data Access – Tarfile-based

The Data Access process is as follows: files are extracted from tar balls of the dataset, we call the *source* in the following discussion, based on results from a search onto a temporary location, we call the *staging area*. Additional filters and post-processors are applied to transform the data into the format useful for the analysis, and the final files are repackaged into a tar archive, and placed into a location, we call the *destination*, to await delivery to the end users. In our case study, the combination of file systems on which the source, staging, and destination are located was subject to variation in order to assess the I/O efficiency of the file systems for use with the NSVS dataset. To simplify the benchmarking process neither filters, nor post-processors were applied. Single threading was used in both the benchmarked extraction and repacking scenarios. The RAID and SSD benchmarks were performed on a private server with no other user on the system, while the GPFS and Lustre

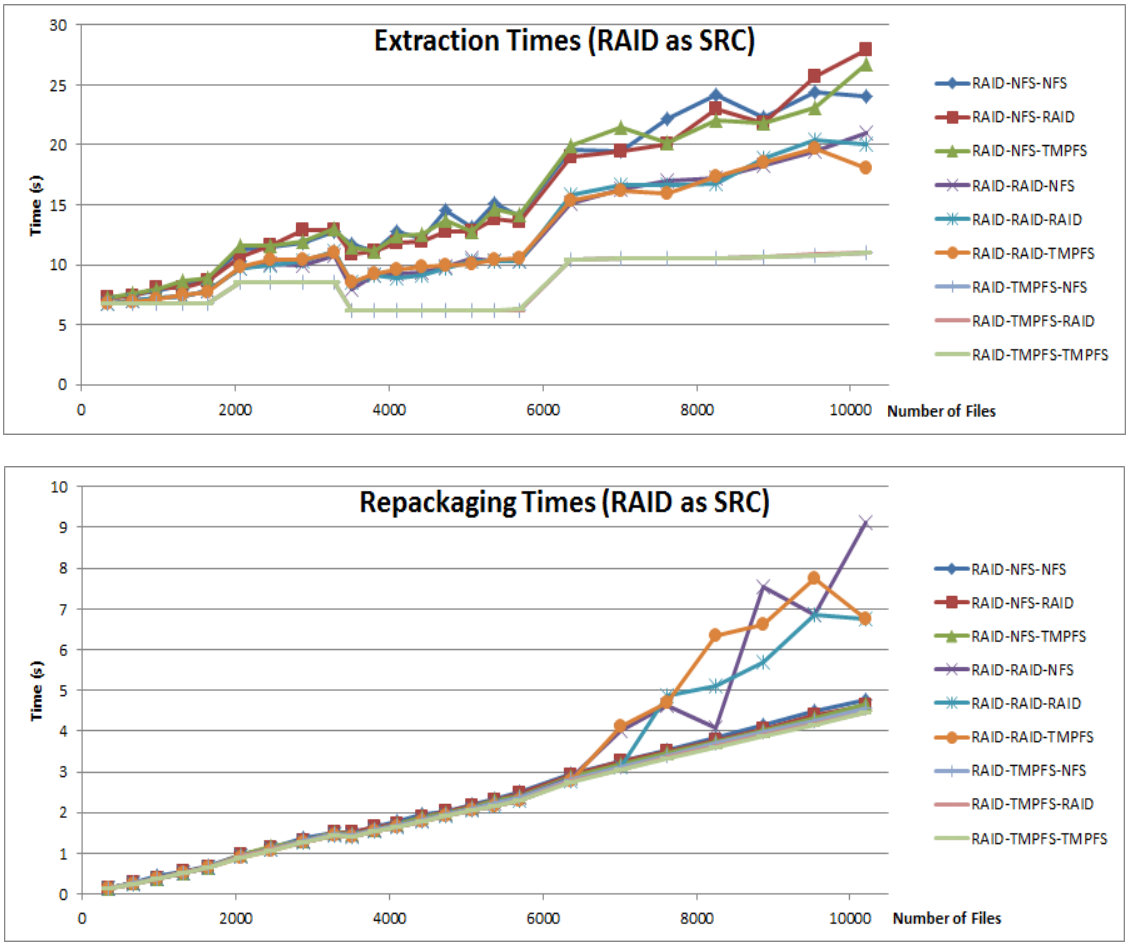


Figure 2: Data access performance (RAID as source)

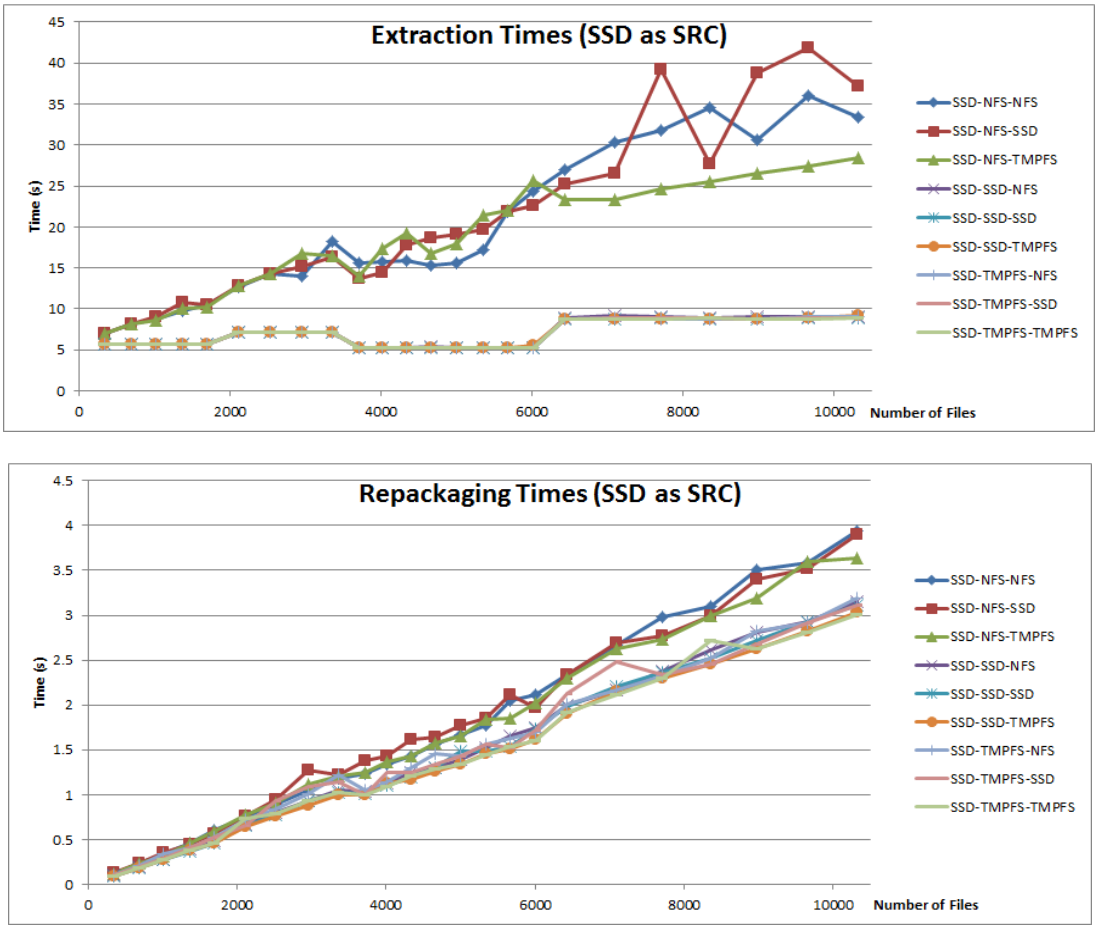
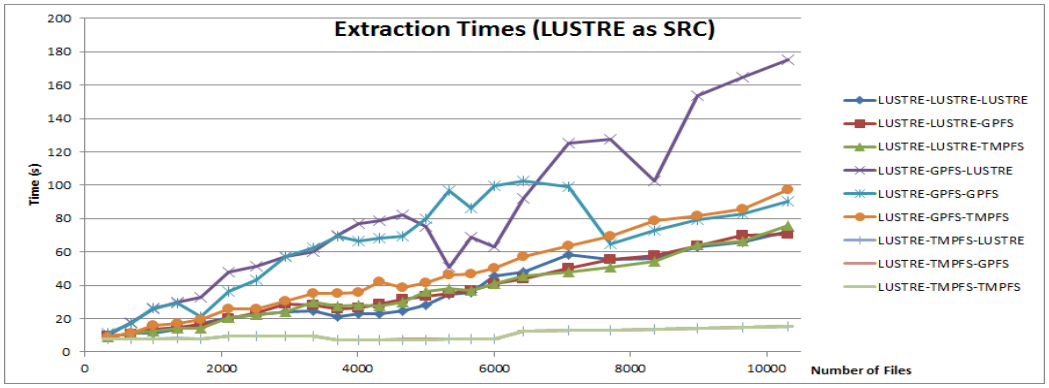


Figure 3: Data access performance (SSD as source)

benchmarks were performed on the Hopper system at NERSC, where there are always tens if not hundreds other users on the system.

From all benchmark scenarios, the actual bottleneck in the Data Access process is not in the results-packaging stage but in the data-extraction; we have shown that data-extraction consistently requires approximately an order of magnitude more time than the packaging of results (figures 2 to 5). The results also indicate that, for this case study, not only does using tmpfs as the staging area greatly improves both the extraction and repackaging times regardless of the file systems used as the source and destination, but it also scales much better linearly with respect to the number of files to be accessed. In terms of file extraction from datasets, it is shown that SSD is the best option for both the reading and writing of many small files (figure 3). Distributed file systems that are optimized for reading and transferring a small number of large files in cluster computing, such as GPFS and Lustre, have been shown to be at a disadvantage in both reading and writing, compared to a simple networked file system such as NFS (figures 4, 5). These benchmarks, however, are raw timings and do not account for the data transfer latencies due to fluctuations in network traffic in the case of the benchmarks on the various distributed/networked file systems. It can be concluded, however, that, given background network latencies, local file systems such as RAID or SSD provide much more consistent I/O rates.



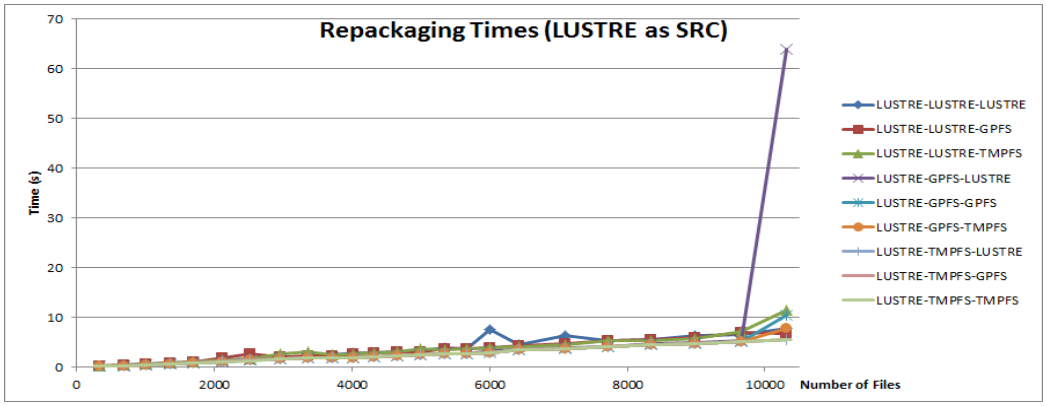
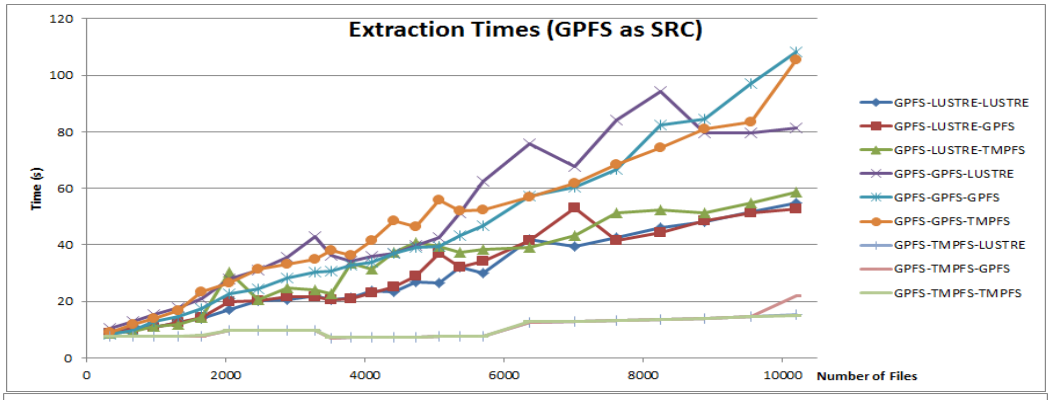


Figure 4: Data access performance (Lustre as source)



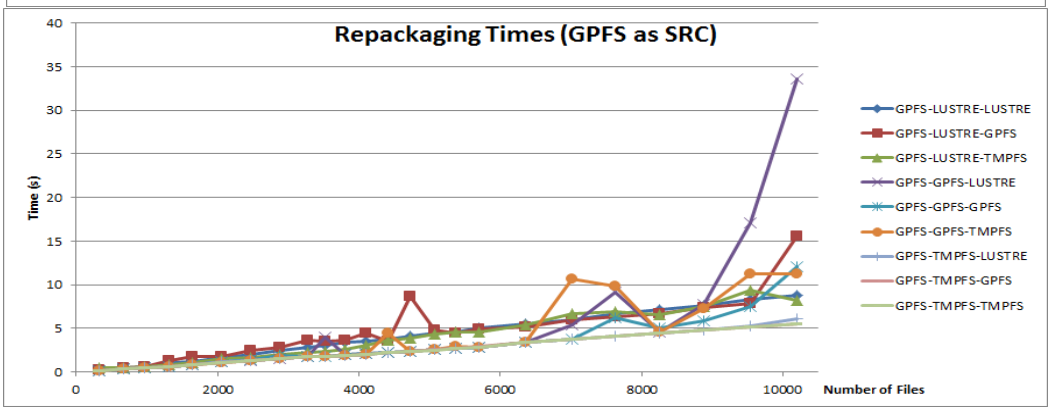


Figure 5: Data access performance (GPFS as source)

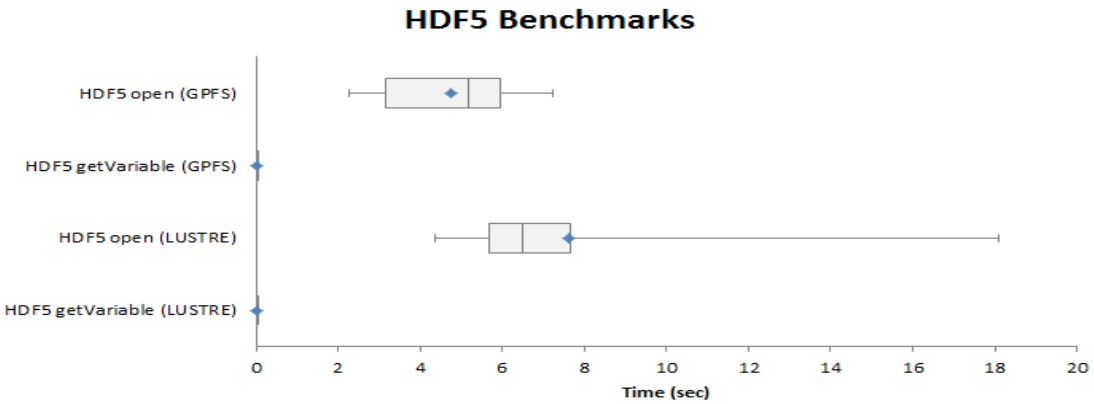


Figure 6: HDF5-based data access performance benchmark

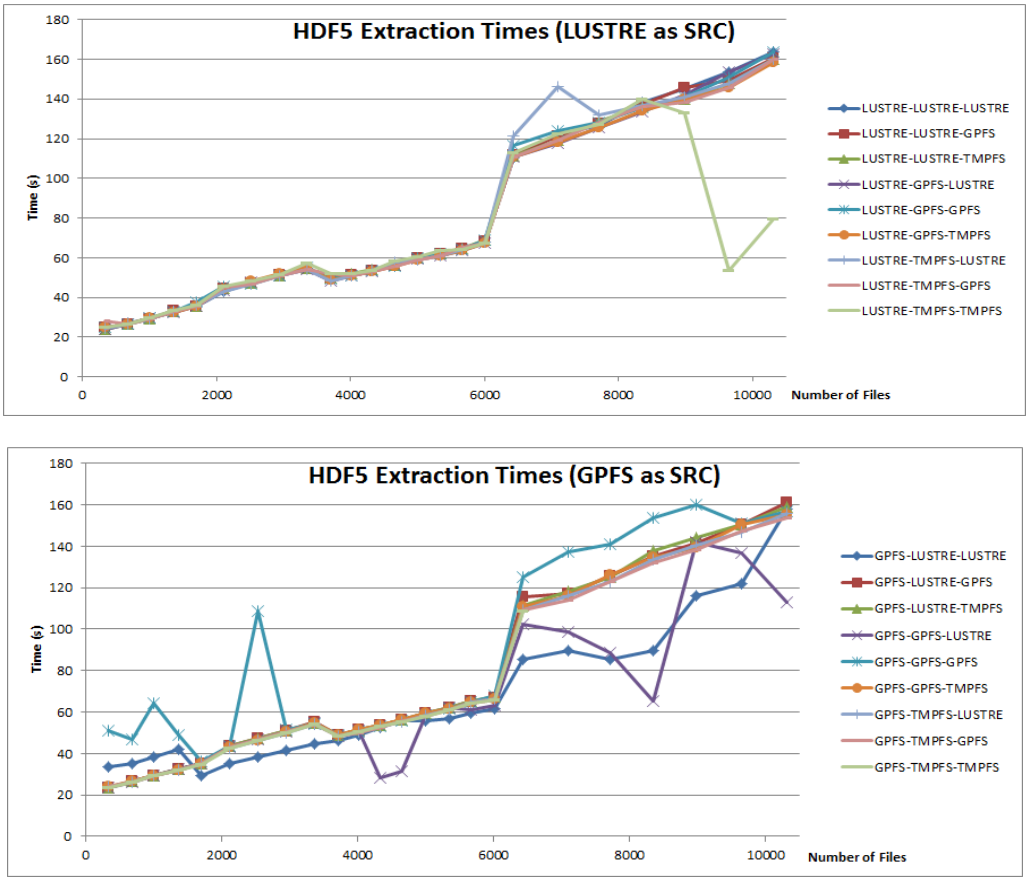


Figure 7: HDF5-based data access performance

D. Data Access – HDF5-based

The Data Access flow for HDF5 is similar to that for the tar ball data-archiving method. In the HDF5 data organization scheme, light curve files are packaged into HDF5 files as arrays, or variables in HDF5 terminology. A subset of light curve files that was organized into a specified tar ball is now a set of variables in the corresponding HDF5 file; thus, the NSVS dataset consists of 638 HDF5 archives. Similar file system benchmarkings were carried out, and to ensure accurate, reproducible timings, a small C++ program was written that called FastQuery's simple HDF5 access API to read data out of the HDF5 files. Because FastQuery is build on FastBit technology, the speed of FastQuery search was not of main concern as it is projected to be on same order as FastBit search, so only FastQuery data extraction benchmarks were carried out. Again, to simplify the benchmarking process neither filters, nor post-processors were applied, and only single threading was used. Both benchmarks were performed on the Hopper system at NERSC.

Based on the benchmark results, the choice of file system for staging (tmpfs or not) had little, if any, impact on improving HDF5 extraction times (figure 7). This is probably because only one file (the HDF5 archive) is being read at a time when multiple light curve datasets are being extracted from it, whereas with the tar ball archive, multiple files are being read from and written to disk. The second observation to note is that on the whole, HDF5 data extraction on average takes much longer time than data extraction from tar balls. The majority of the time is actually spent on opening and reading in the HDF5 files by FastQuery; opening and loading an HDF5 file into memory costs an average of 4.76 and 7.64 seconds for HDF5 archives stored on GPFS and Lustre file systems, respectively, while reading out individual variables of an HDF5 archive took a mere average of 1.73E-04 and 6.52E-04 seconds for HDF5 archives stored on GPFS and Lustre file systems, respectively (figure 6). This is because upon opening the HDF5 file, FastQuery caches the archive for later use. Because the NSVS dataset consists of 638 HDF5 archives, one Data Access procedure may require opening many HDF5 archives, which significantly increases extraction time. One solution to address this problem is to merge the dataset into a smaller number of HDF5 archives to reduce the number of HDF5 file open calls, but its benefits and tradeoffs, especially in caching and memory usage, have not been thoroughly assessed, and will be explored in future work.

V. SUMMARY

In this paper, we have described the design of the Attribute-based Unified Data Access Service and the performance of the data access for an astronomy data set. The purpose of this case study is to investigate methods to reduce data to be transported across networks in order to optimize network transport and minimize network latencies in cloud-based scientific data analyses.

This study shows that FastBit is a much better alternative database indexing method to indexing through catalog text file. FastBit has proven to be very scalable for datasets with high file set cardinalities. Under a Cloud Computing environment that is supported by reasonably high-throughput networking, with FastBit in place for dataset querying, the bottleneck in any data analysis thus lies in data access process. In our case study, the extraction of data from the NSVS dataset takes on average two orders of magnitude more time than the search operation, and the repackaging and compressing of resulting takes on average one order of magnitude more time than the search operation. A standard source-staging-destination model was recognized for the data access in the Cloud Computing environment, in which a subset of the dataset located at a source is extracted to the staging area, post-processed, and repackaged into a destination folder for network delivery. Studies indicate that using tmpfs as the file system for the staging area resulted in a substantial overall data access time reduction. Benchmarks with high-performance distributed file systems such as GPFS and Lustre indicate that they are far from optimal in scenarios such as this case study, in which many small files are being moved around, because they are file systems optimized for reading and moving very large files. Both tar ball-based and HDF5-based benchmarks using GPFS file systems gave less "stable" timing measurements, with much more I/O time variability. Whether this is because the GPFS system in our experiment was mounted on a very busy node or because this is a side effect of GPFS's design and implementation remains to be ascertained.

From the benchmark results, it can be inferred that tmpfs is both the ideal choice of file system in general for reading and writing small files, with SSD coming slightly behind in both. While RAID performs better than NFS for writes, it appears that NFS only slightly outperforms RAID in read rates, although both are outperformed in both aspects by SSD.

On the HDF5 benchmarks, it was demonstrated that the majority of the HDF5 reading time was spent on loading the HDF5 file, not retrieving the data within the HDF5 archive.

Because an HDF5 archive appears as one large file in the file system, as opposed to many files in the case of tarfiles, there are significantly fewer system calls for opening files, both for reading and writing, and so the data extraction times are largely independent of the type of file system the source dataset was mounted on. Furthermore, in the case of using HDF5 archives, utilizing tmpfs as staging provided little, if any, speed improvement. Because of the time used to open and cache in HDF5 archives to memory, HDF5 data extraction on the whole is slower than tarfile-based data extraction. In the case of the NSVS dataset, merging the HDF5 files into a smaller file set may be a viable option to avoid this, but requires re-indexing and redefining the parameters for the Field_ID attribute, and there could potentially be tradeoffs with cache and memory usage.

Future work involves further refinement of the data access process to ease the load on the data delivery (network transport) process. Since the aforementioned benchmarks only ran on single-threaded mode, multi-threading will be explored in future work. Preliminary multithreading benchmarks have been shown to increase tarfile-based data extraction rate by at least 2-3x. As both FastBit and FastQuery support multithreading mode, future work will entail assessing possible improvements with multithreaded querying.

The current HDF5 benchmarks used FastQuery to open into the dataset to retrieve the data, not query the dataset. Future work will include loading only the catalog index file for data querying by FastQuery, and use the HDF5 library directly to access the bulk of the dataset. Finally, alternative data packaging formats will be tried, such as netCDF, which is also supported by FastQuery, or other compression formats that might enable either smaller files or faster decompression times, such as 7z or BZIP.

ACKNOWLEDGMENTS

This work was funded by Korea Institute of Science and Technology Information under the agreement C12022 at KISTI and WF008977 at LBNL. This work also used resources of the Lawrence Berkeley National Laboratory and National Energy Research Scientific Computing Center, by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under

contracts DE-AC02-05CH11231. Y.I.B. acknowledges the support of the National Research Foundation of Korea through grant 2012-0009094.

REFERENCES

[1] Sloan Digital Sky Survey Data Release 7: http://www.sdss.org/dr7/

[2] SuperWASP public data archive Data Release 1: http://www.wasp.le.ac.uk/public/

[3] Northern Sky Variability Survey (NSVS), http://skydot.lanl.gov/nsvs/nsvs.php

[4] "Northern Sky Variability Survey: Public Data Release", P. R. Woźniak et al. 2004 The Astronomical Journal, 127 2436, doi:10.1086/382719

[5] “FastBit: Interactively Searching Massive Data”, K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. C. Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Lauret, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, O. Rubel, A. Shoshani, A. Sim, K. Stockinger, G. Weber, and W.M. Zhang, In Proc. SciDAC 2009.

[6] "Parallel Index and Query for Large Scale Data Analysis", J. Chou, K. Wu, O. Rubel, M. Howison, J. Qiang, Prabhat, B. Austin, E. W. Bethel, R. D. Ryne and A. Shoshani, IEEE Supercomputing Conference (SC), November 2011.

[7] “GridFTP: Protocol Extensions to FTP for the Grid”, W. Allcock, et all. Global Grid Forum, GFD.020, April, 2003.

[8] “Detrending time series for astronomical variability surveys”, D.W. Kim, P. Protopapas, C. Alcock, Y.I. Byun, and F.B. Bianco, 2009, Monthly Notices of the Royal Astronomical Society, 397, 558

[9] “Improved Time-Series Photometry and Calibration Method for Non-Crowded Fields : MMT Megacam and HAT-South Experiences”, S.W. Chang, Y.I. Byun, and D.W.Kim, 2012New Horizons in Time-Domain Astronomy, Proceedings of the International Astronomical Union, IAU Symposium, Volume 285, p. 291-293